Network Programming:
Ch.2 Transport Layer: TCP and UDP

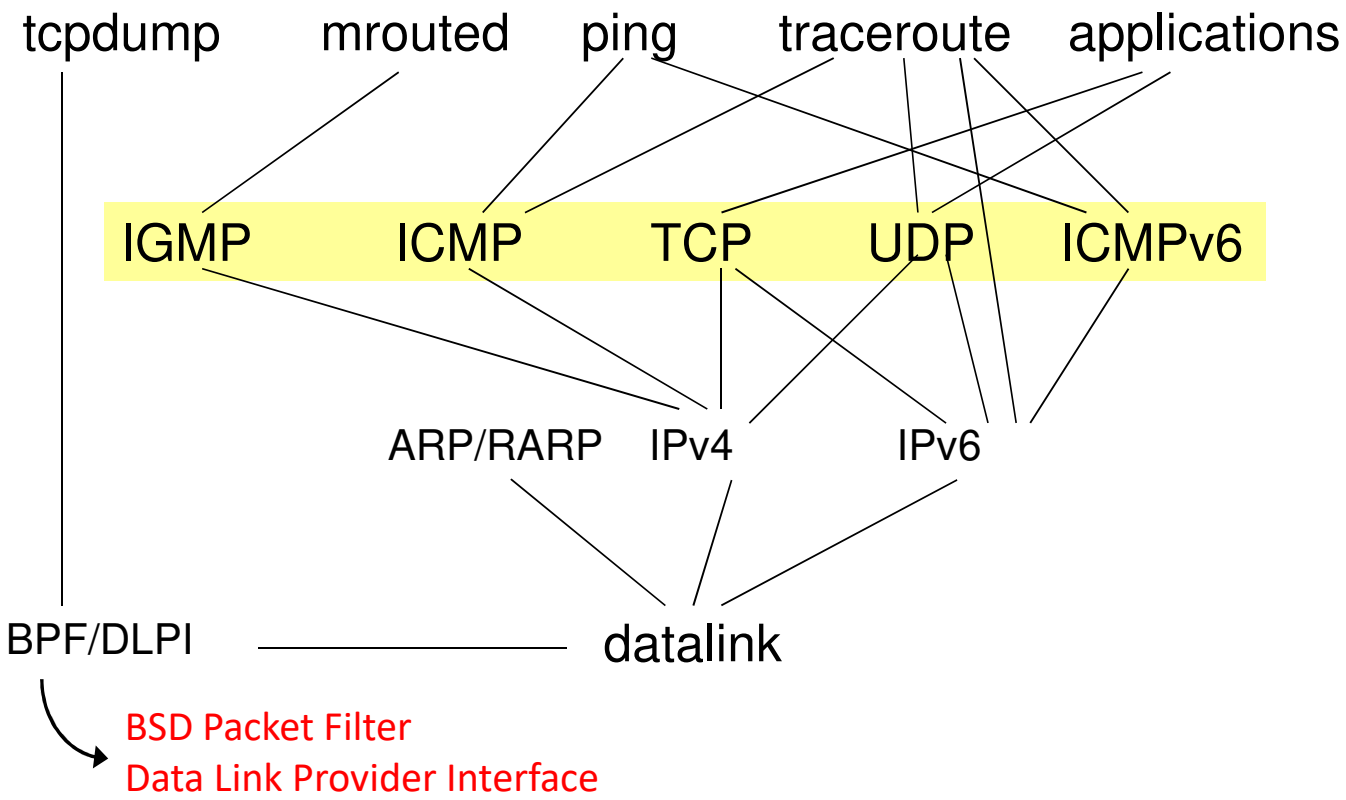
Li-Hsing Yen
NYCU
Ver. 1.0.1

Transport Layer: TCP and UDP

- Overview of TCP/IP protocols
- Comparing TCP and UDP
- TCP connection: establishment, data transfer, and termination
- Allocation of port numbers
- Size matters: MTU, datagram, MSS, buffer
- Standard Internet services and applications
- Debugging techniques and tools

Some materials in these slides are taken from Prof. Ying-Dar Lin with his permission of usage

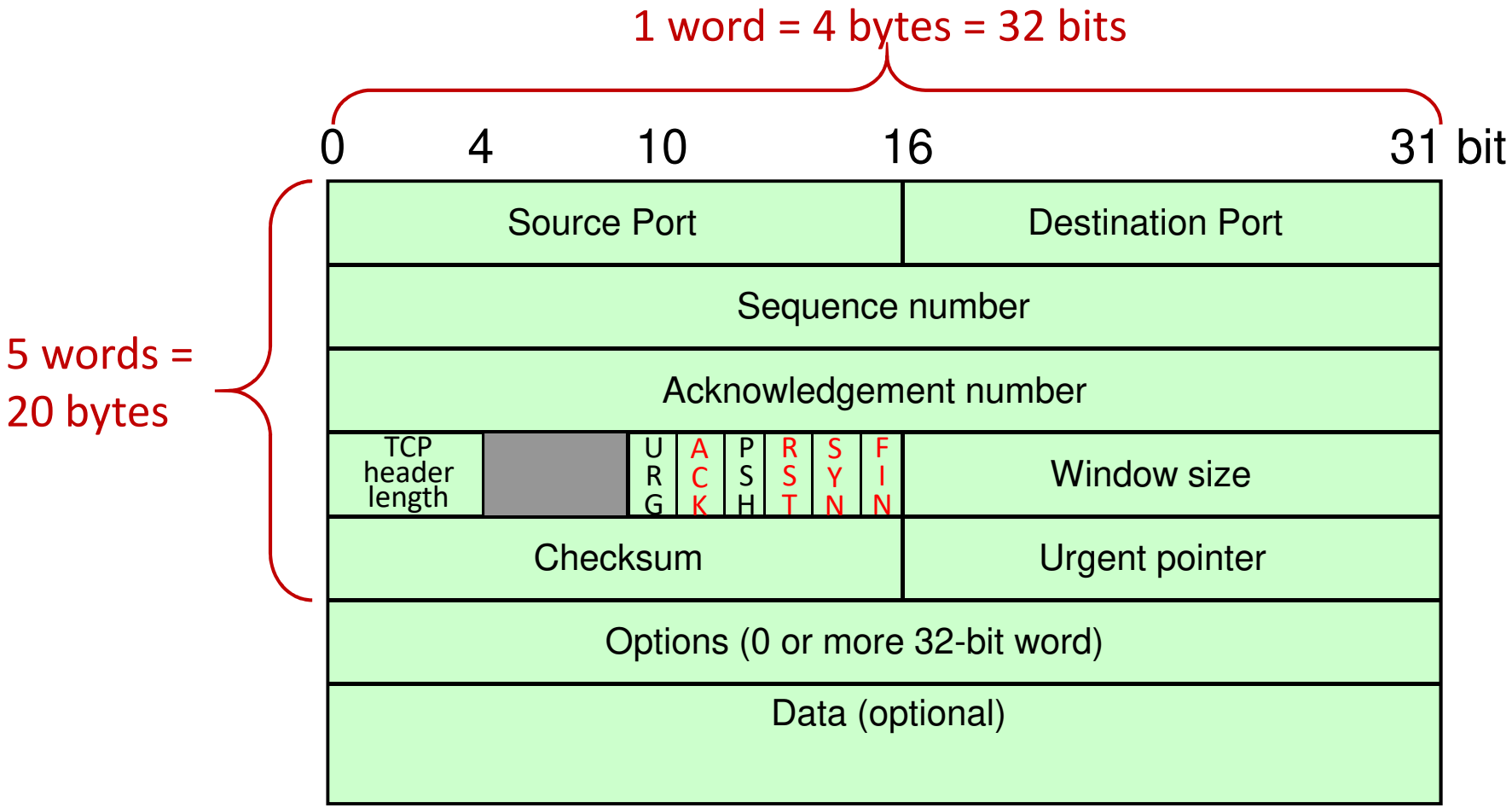
Overview of TCP/IP Protocols



Comparing TCP and UDP

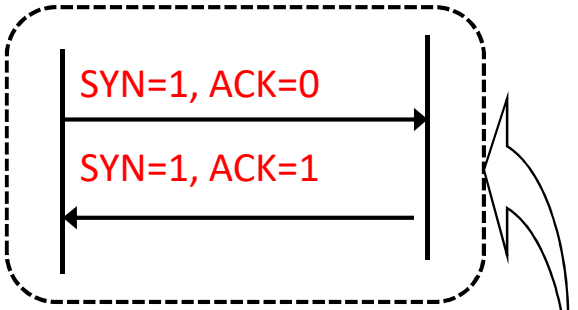
	TCP	UDP
Binding between client and server	Yes (connection-oriented)	No (connection-less)
Data	Byte-stream	Record
Reliability	Yes (ack, time-out, retx)	No
Sequencing	Yes	No
Flow control	Yes (window-based)	No
Full-duplex	Yes	Yes

TCP Segment Header



Some TCP Header Fields

- Sequence number (32 bits)
  - actually the byte number
- Acknowledgement number
  - specifies the next byte expected
- Flags



URG	1 if the Urgent pointer is in use	RST	To reset a connection
ACK	the Acknowledgement number is valid	SYN	To denote connection request and accepted
PSH	This segment requests PUSH	FIN	To release a connection (in one direction)

More TCP Header Fields

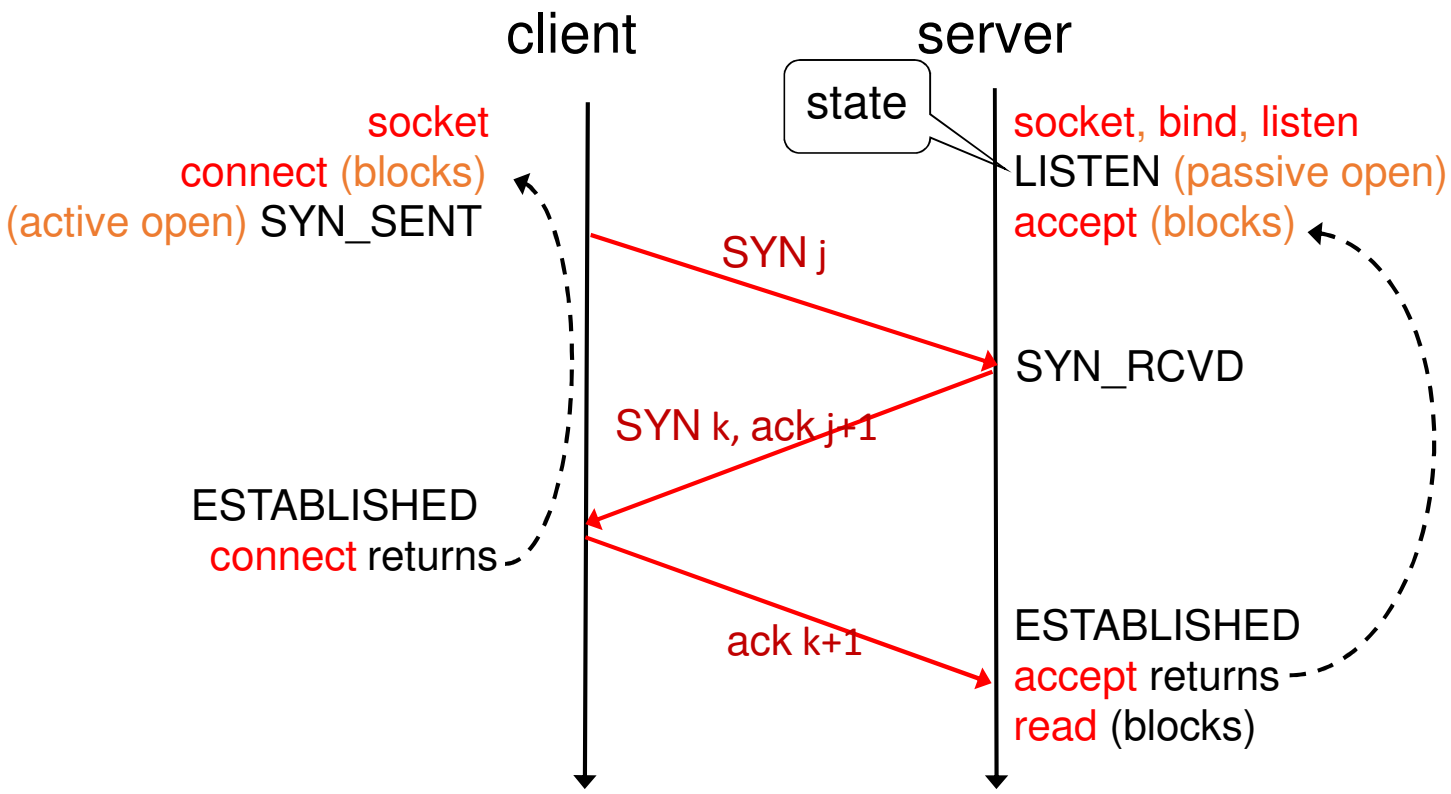
- Window size (16 bits)
  - tells how many bytes may be sent starting at the byte acknowledged (receiver's window size)
- Urgent pointer (16 bits)
  - indicate a byte offset from the current sequence number at which urgent data are to be found
  - urgent data: when an interactive user hits the DEL or CTRL-C key, the sending application puts some control information in data stream

TCP Options

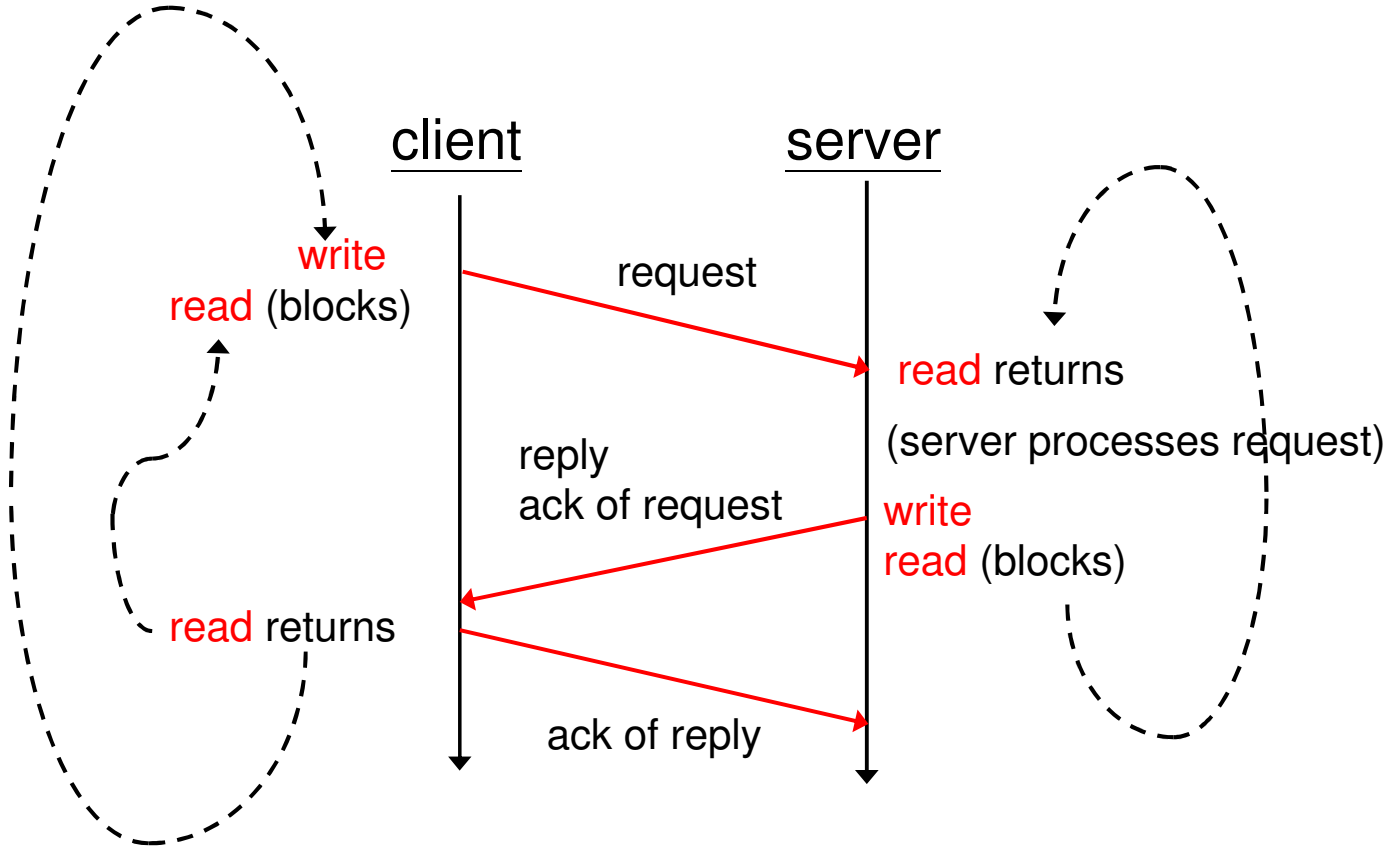
- MSS Option: maximum segment size
- Window scale: (new)
  - The maximum window size that either TCP can advertise to the other is 65535
  - This option shifts the advertised window by 0-14 bits (resulting in a maximum of 1GB)
- Timestamp: (new)
  - Prevent possible data corruption

TCP Connection: Establishment

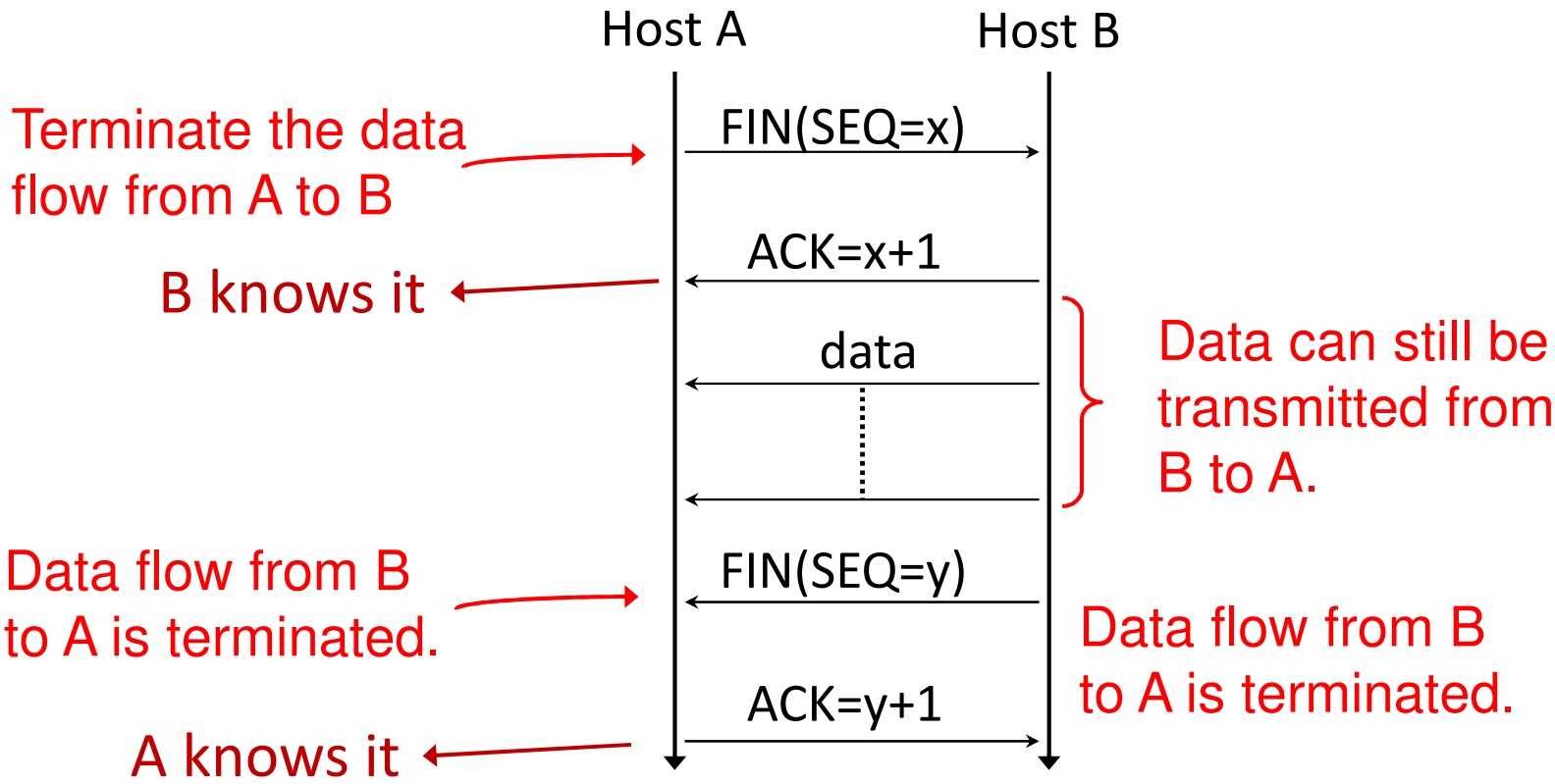
Three-way handshake



TCP Connection: Data Transfer

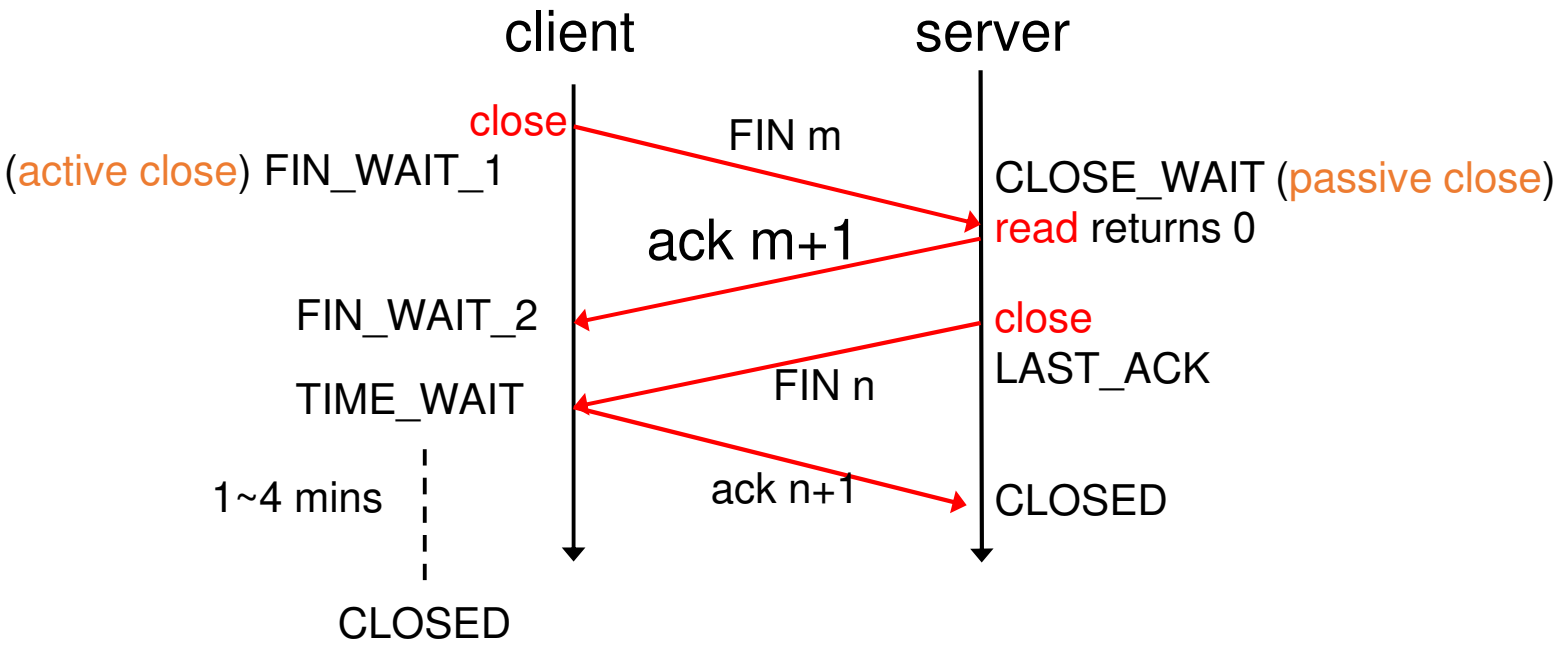


TCP Connection Release

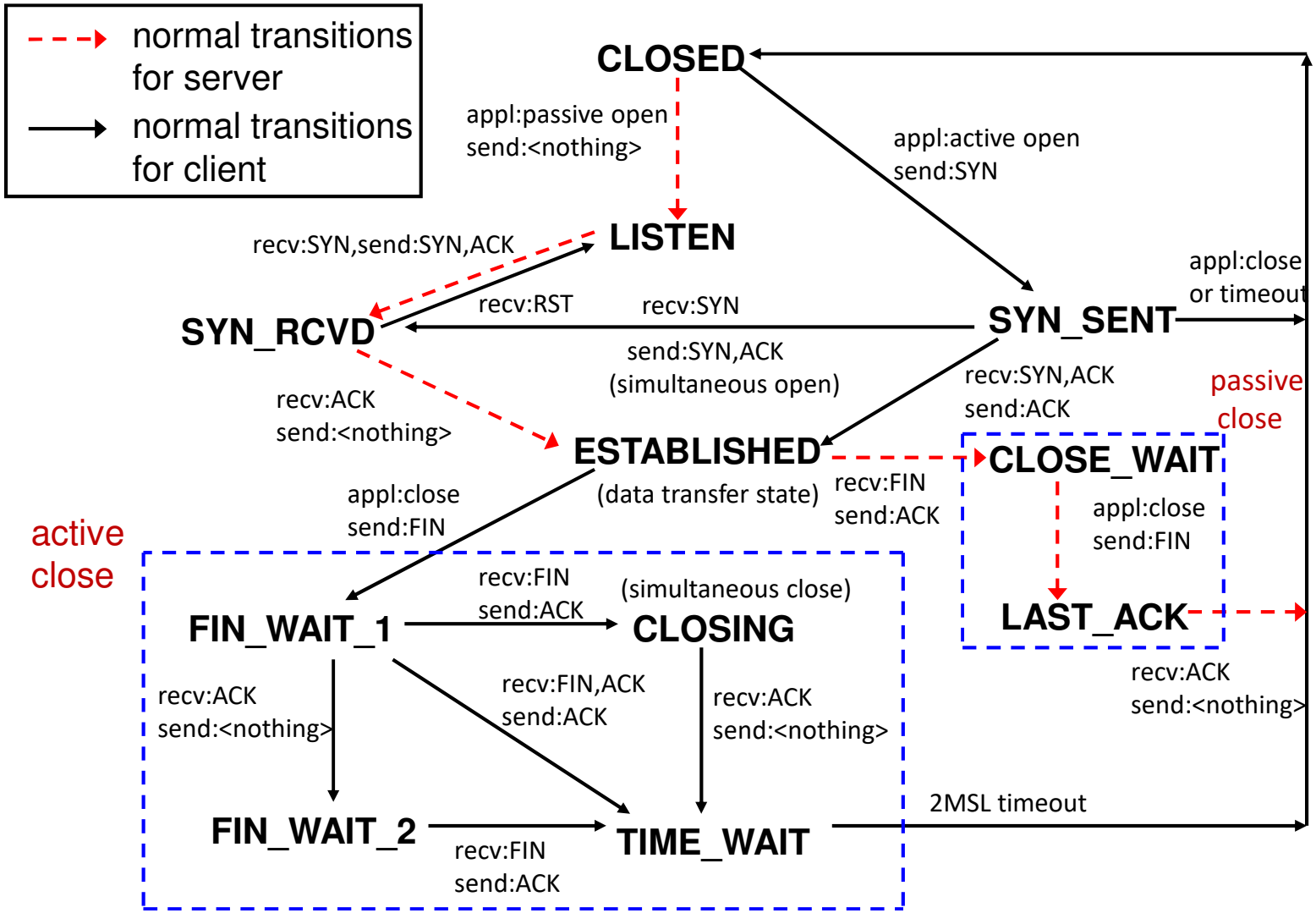


TCP Connection: Termination

Four-way handshake



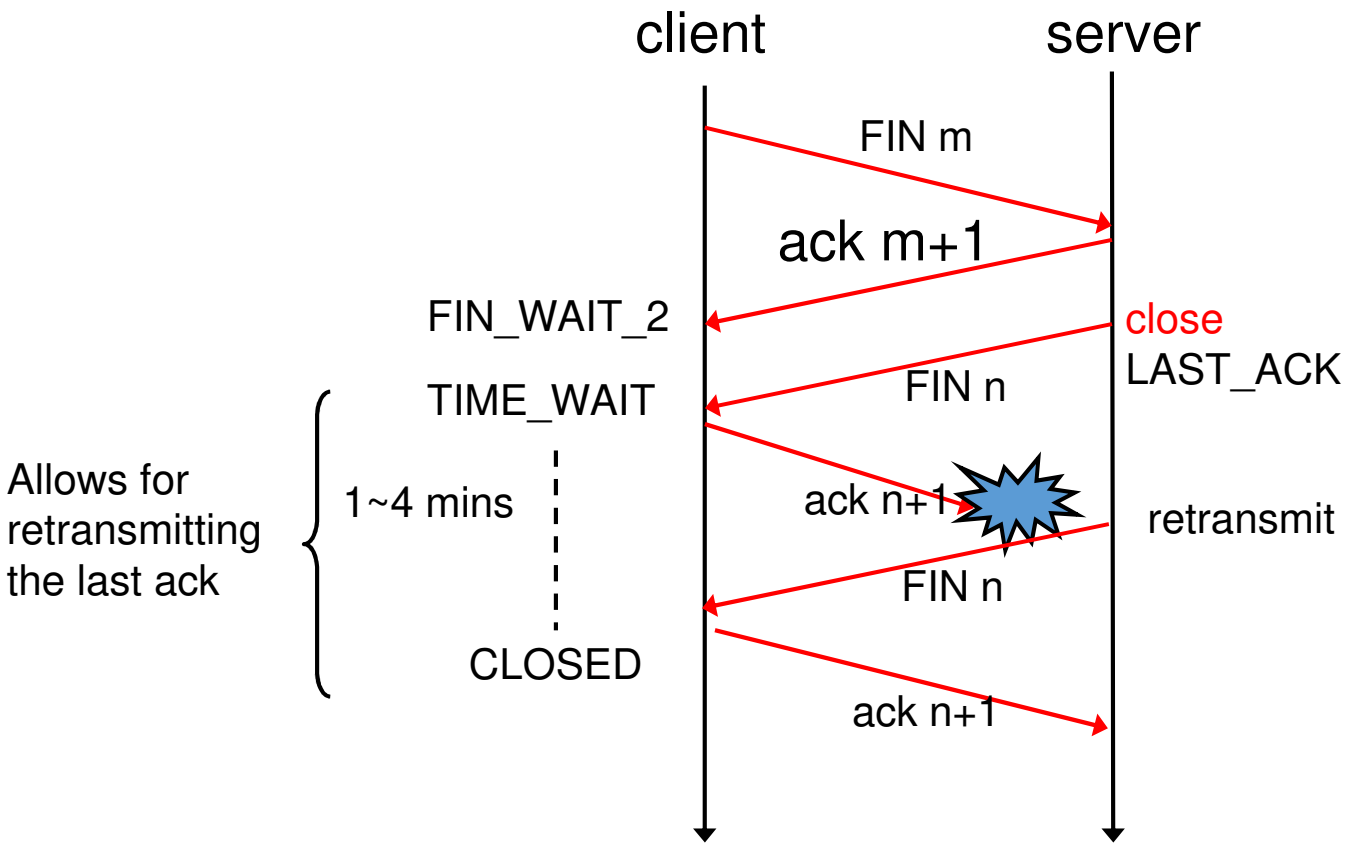
TCP State Transition Diagram



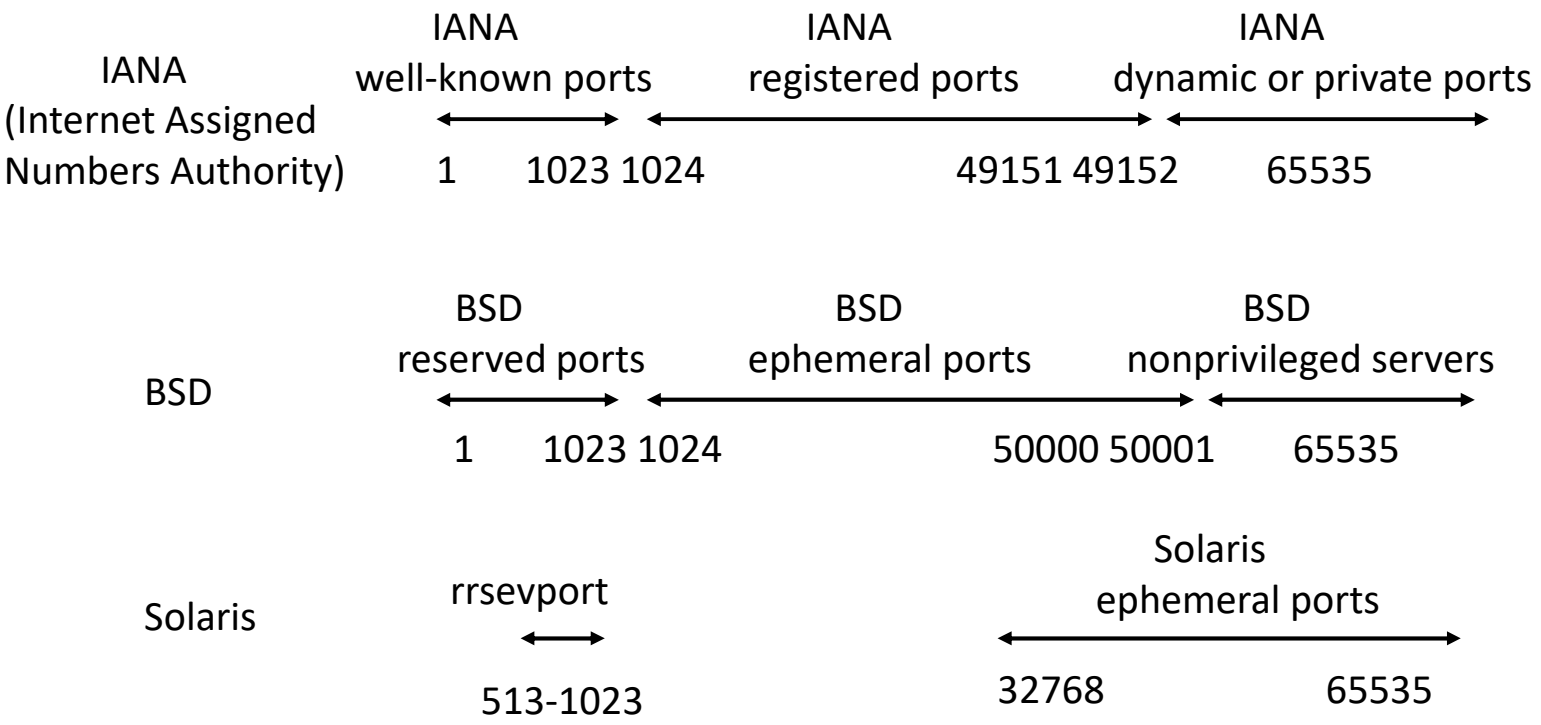
TIME_WAIT State

- Two reasons
  - To implement TCP's full-duplex connection termination reliably
    - The final ACK might need to be retransmitted
  - To allow old duplicate segments to expire in the network
    - Otherwise old duplicates will be misinterpreted as belonging to a new incarnation (having the same IP/port pair as the terminated one)

Illustrating the First Reason



Allocation of Port Numbers

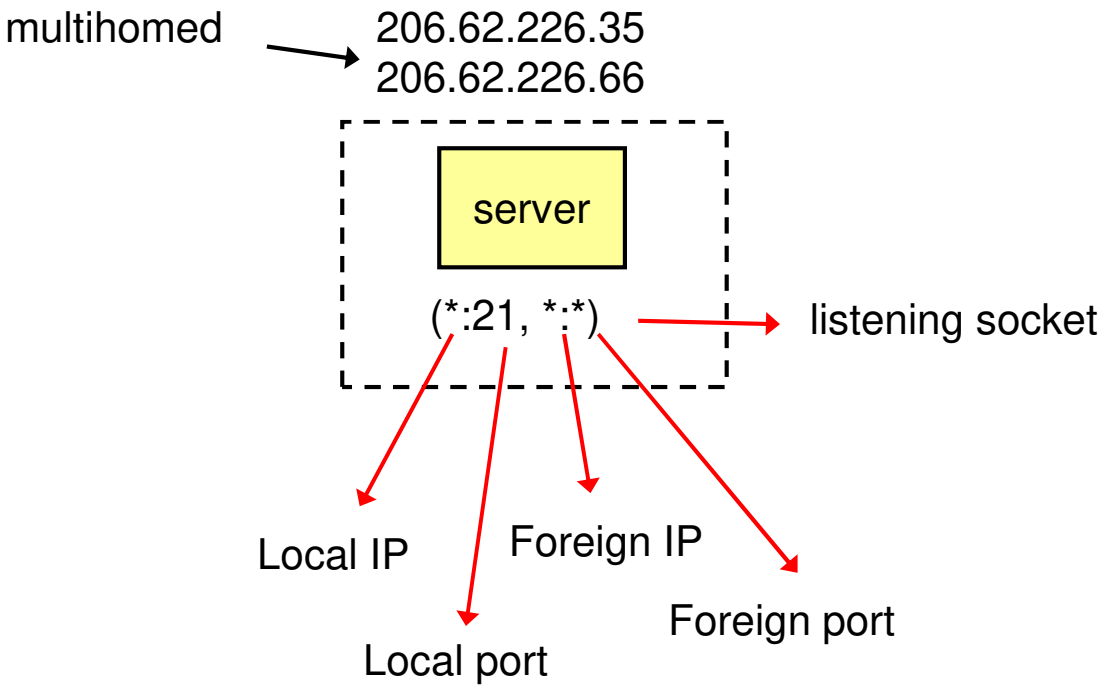


Socket Pair

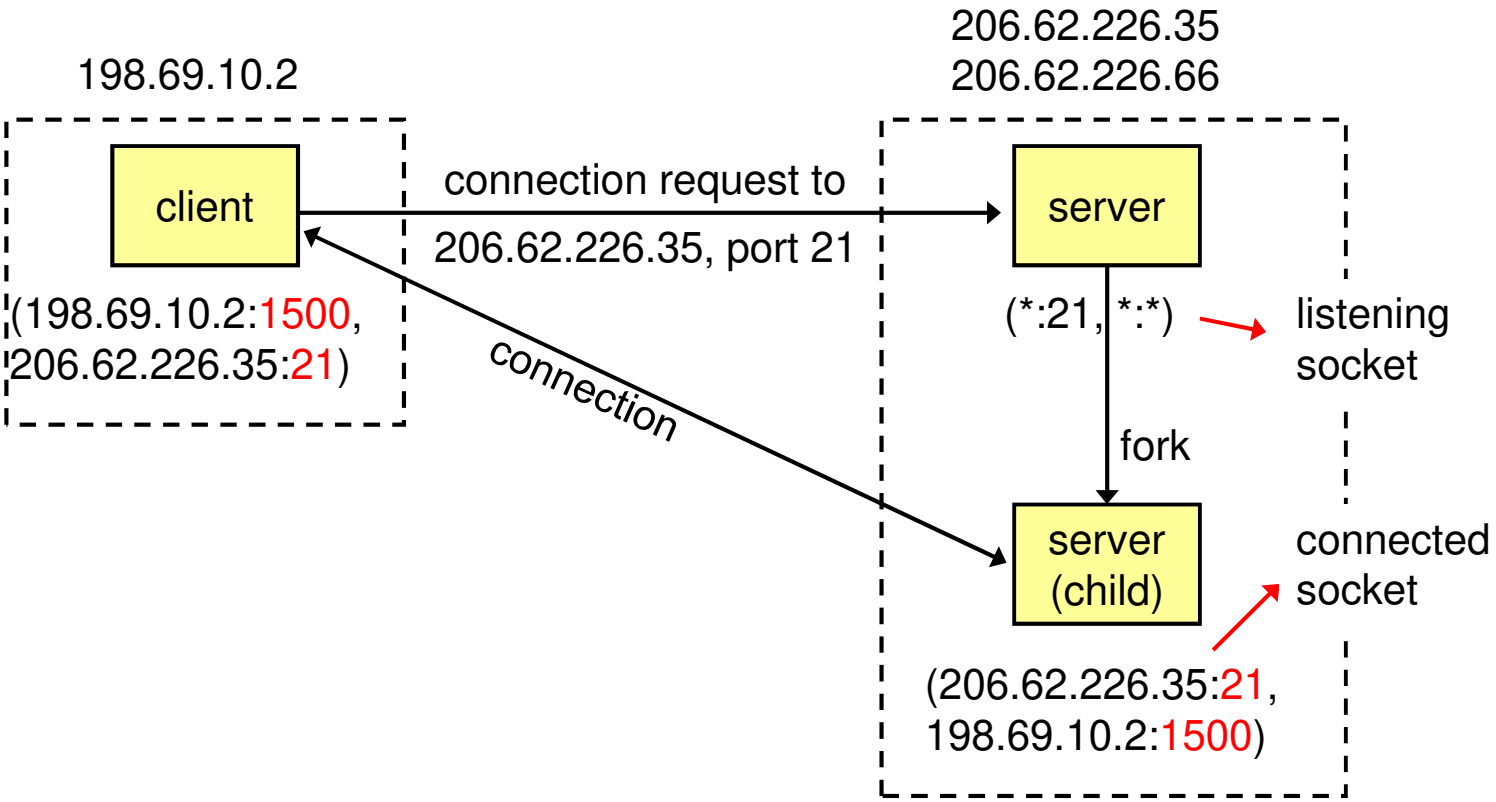
- 4-tuple that defines the two endpoints of a connection
  - The local IP address, local TCP port, foreign IP address, and foreign TCP port

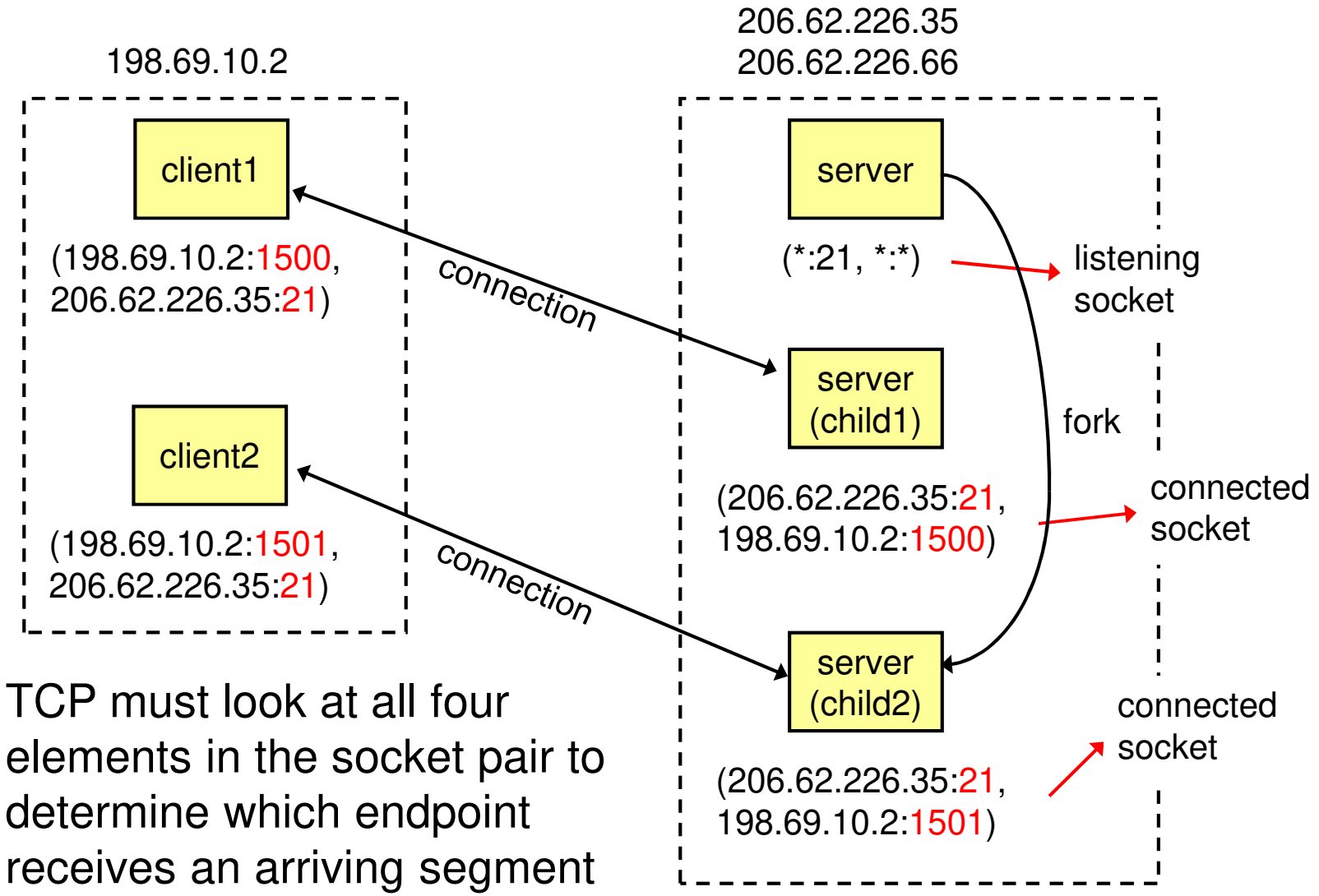
- Uniquely identifies every TCP connection on the Internet

A Concurrent Server



Connect Request from Client to Sever

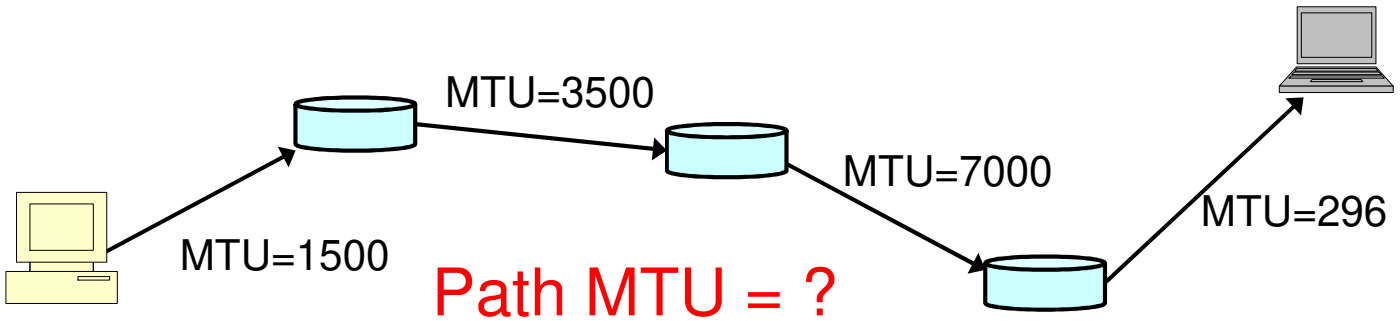




TCP must look at all four elements in the socket pair to determine which endpoint receives an arriving segment

Buffer Sizes and Limitations

- *Link* MTU (maximum transmission unit): Ethernet MTU: 1500 bytes, PPP MTU: configurable
- *Path* MTU: the smallest link MTU in the path, can be discovered by IPv4 DF (don't fragment) bit



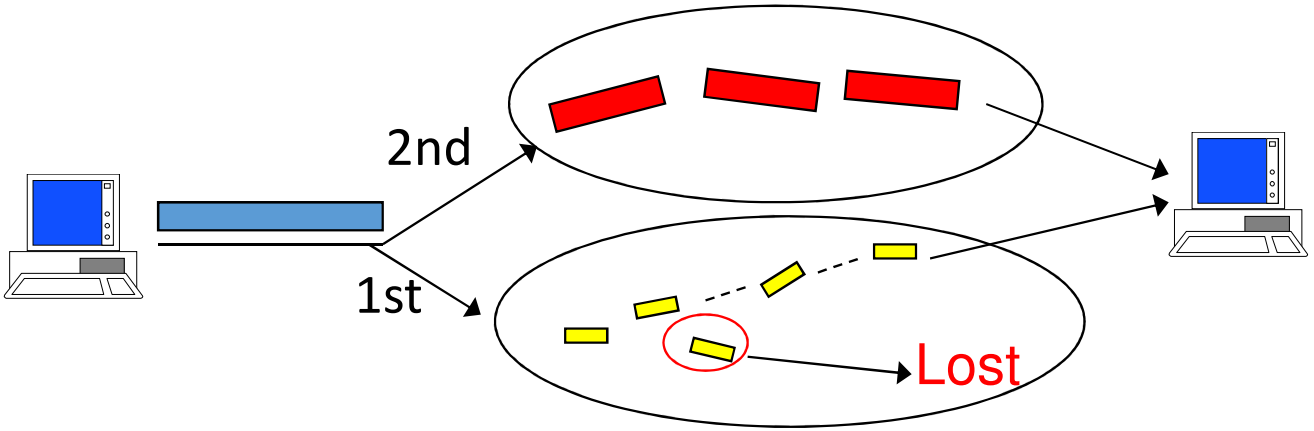
Fragmentations

- Cut datagrams into fragments
- Performed when size of datagrams > link MTU
- Fragments are reassembled only at the final destination
- IPv4 DF bit tells routers not to fragment this datagram
  - May get ICMP “destination unreachable, fragmentation needed but DF bit set” error message

TCP Fragmentation

- A segment will be fragmented when it passes through a transit network with a smaller MTU

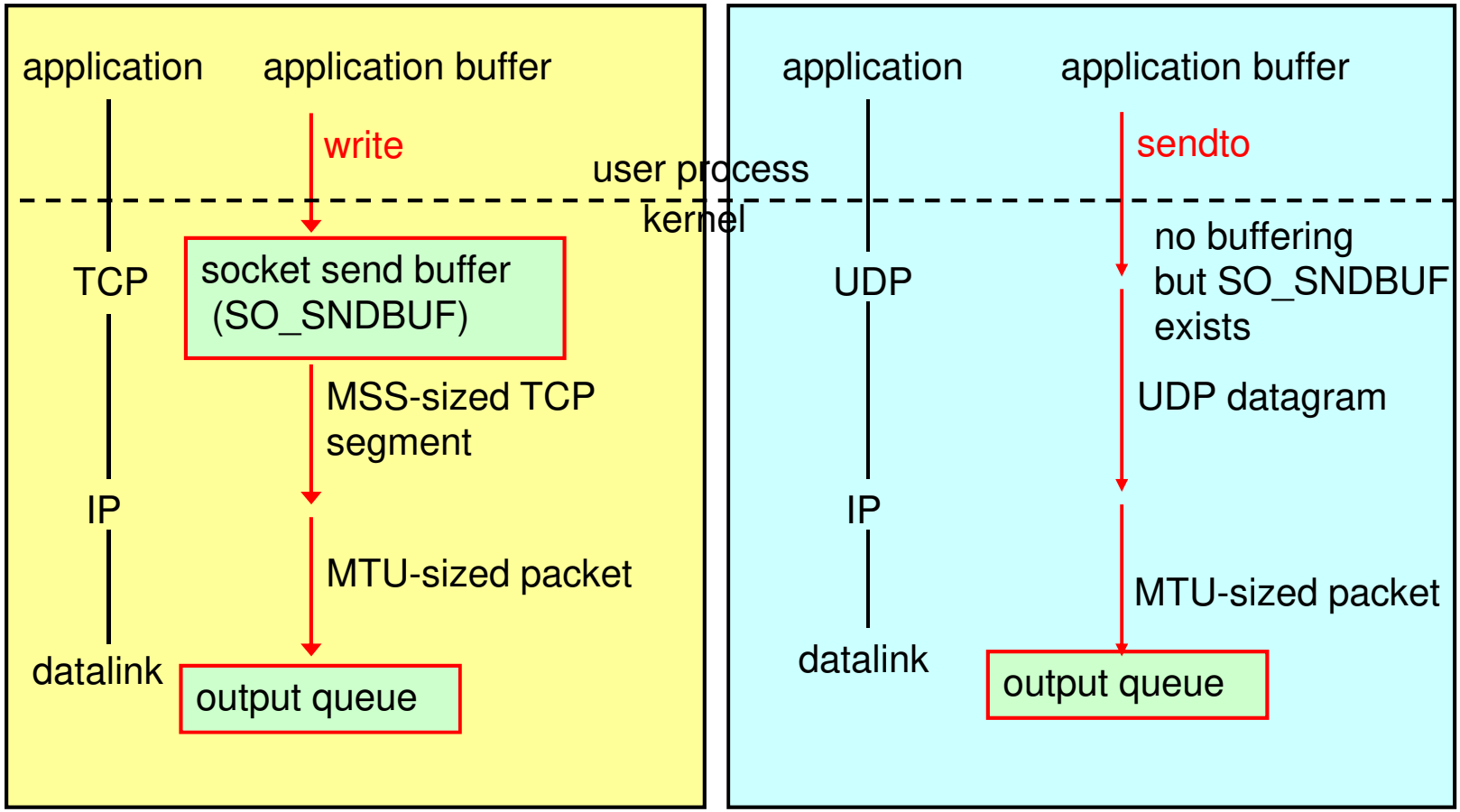
Problem with fragmentation



Maximum and Minimum

- Maximum IP datagram: 65535 (IPv4), 65575 (IPv6) (IPv6 has 32-bit jumbo payload option)
- minimum IP reassembly buffer size: that we are guaranteed any implement must support
- TCP MSS (maximum segment size): actual value of reassembly buffer size, often the link MTU minus IP and TCP headers, to avoid fragmentation

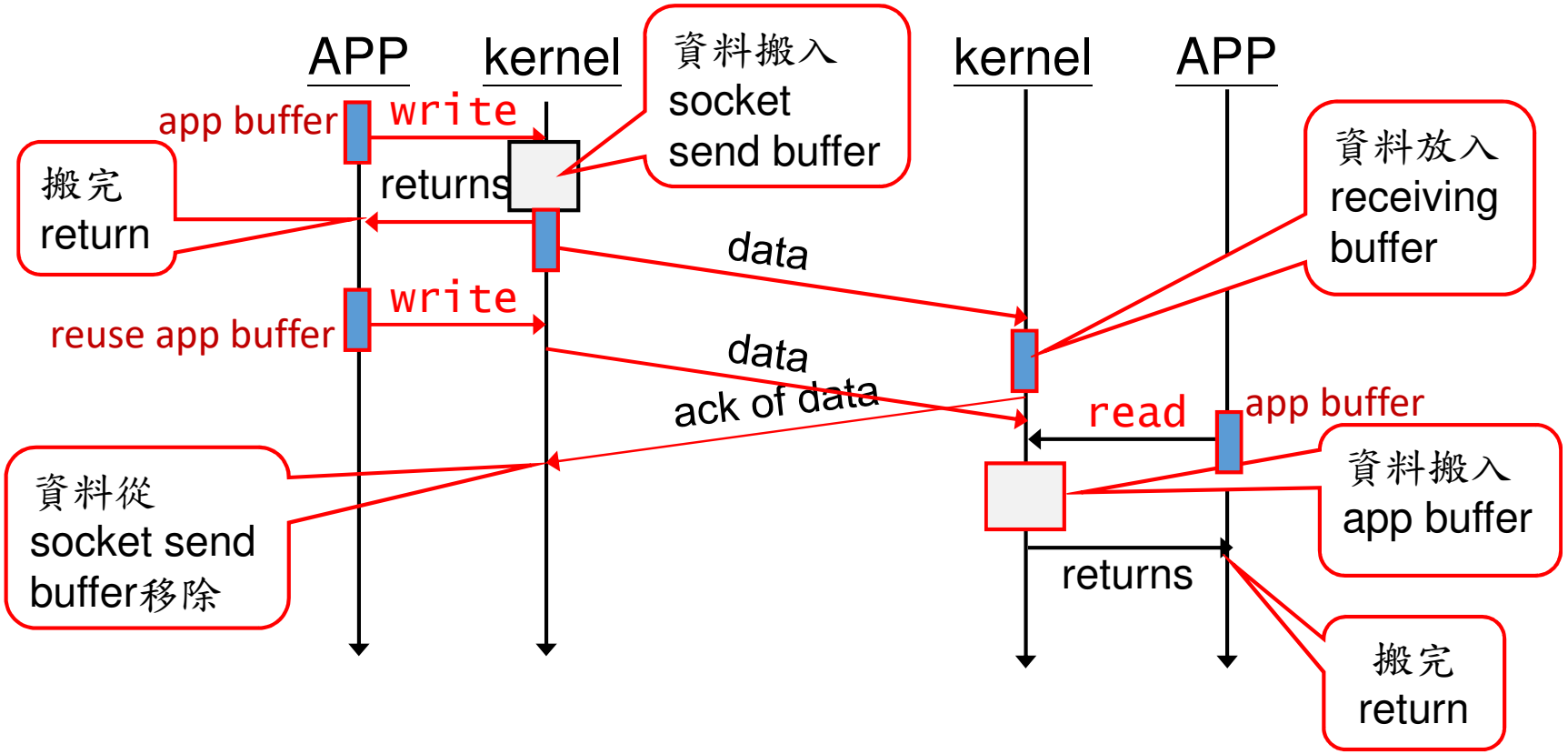
TCP Output and UDP Output



TCP Output

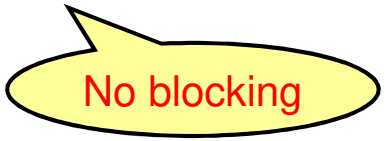
- The write blocks if no room in the socket send buffer
- The kernel will not return from the write until the final byte in the application buffer has been copied into the socket send buffer
- Successful return from a write only tells us that we can reuse our application buffer
- TCP (kernel) keeps a copy of our data until it is acknowledged by the peer

TCP Output Illustrated



UDP Outputs

- UDP socket has a send buffer size but no real buffer
- If application's datagram > send buffer size, an error message is returned
- UDP need not keep a copy of our data
- A successful return from a write to a UDP socket tells us the datagram has been added to the data link output queue
- An error message will be generated if there is no room in the data link output queue



Standard Internet Services and Applications

- Standard services provided by *inetd* daemon: echo/port7/RFC862, discard/port9/RFC863, daytime/port13/RFC867, chargen/port19/RFC864, time/port37/RFC868
- tested by “telnet machine service”, service mapped by /etc/services
- Common application types: diagnostic, routing protocol, datagram, virtual circuit, etc.

Protocol Usage of Various Common Applications

Application	IP	ICMP	UDP	TCP
Ping		x		
Traceroute		x	x	
OSPF	x			
RIP			x	
BGP				x
BOOTP			x	
DHCP			x	
NTP			x	
TFTP			x	
SNMP			x	
SMTP				x
Telnet				x
FTP				x
HTTP				x
NNTP				x
DNS			x	x
NFS			x	x
RPC			x	x

Debugging Techniques and Tools

- System call tracing: *truss* (in SVR4), *ktrace* & *kdump* (in BSD) (Note that socket is a system call in BSD, while putmsg and getmsg are the actual system calls in SVR4)
- *sock* developed by W.R. Stevens: used to generate special case conditions, as stdin/stdout client, stdin/stdout server, source client, sink server
- *tcpdump*: dump packets matching some criteria
- *netstat*: status of interfaces, multicast groups, per-protocol statistics, routing table, etc.
- *lsof* (list open files): which process has a socket open on a specified IP address or port